Extended Kalman Filter

Shubhankar Kulkarni

October 27, 2022

1 Introduction

The report discusses the application of the **Extended Kalman Filter** for tracking problems where the measuring sensors cannot be equated with linear matrices with the states to be monitored of the object being tracked. In such scenarios, the traditional Kalman Filter that relies upon linear matrix equations fails to give results.

For such problems, the system is described using nonlinear equations for the state transition and measurement equations and is then linearized by the usage of Taylor's expansion for computing the value of the quatity being measured.

This filter is of significance since the majority of the real-world examples of tracking we observe are non-linear in nature and for using the Kalman Filter, they need to be linearized. This involves a degree of abstraction in the way the system is behaving and can result in seeping in of errors in the system model. Extended Kalman Filter proves better at handling such cases with limited abstraction.

2 Methods

The Extended Kalman filter follows the predict and update model of the Kalman Filter and computes the Kalman Gain in every iteration to be added to the update of the state at next stage using the measurement data and the prediction from the previous state.

The main difference is the usage of linear matrices in the Kalman Filter and Jacobians by the Extended Kalman Filter. In essence, an Extended Kalman Filter uses Jacobians to linearize the non-linear system on which then a version of Kalman Filter can be applied. The state transition and measurement equations can be written as follows respectively-

$$x_t = f(x_{t-1}, a_t) \tag{1}$$

$$y_t = g(x_t, n_t) \tag{2}$$

where a_t and n_t represent the dynamic and measurement noises respectively.

x_t and x_{t-1} represent the state variable vectors at $(n+1)^{th}$ and n^{th} instances respectively and y_t represents the measured quantities vector. A computable approximation of the above nonlinear state transition equation can be written through Taylor series expansion as-

$$x(t+\Delta t) = x(t) + \Delta t\dot{x}(t) + \frac{(\Delta t)^2}{2!}\ddot{x}(t) + \frac{(\Delta t)^3}{3!}\dddot{x}(t) + ... \tag{3}$$

The above equation can be approximated as follows for smaller values of Δ t and higher derivatives of x,

$$x_t \approx \tilde{x}_t + \frac{\partial f}{\partial x}(x_{t-1} - x_{t-1,t-1}) + \frac{\partial f}{\partial a}a_t \tag{4}$$

The partial derivatives contained in the above modified state transition equation are known as Jacobians. A Jacobian can be defined as a matrix of the partial derivatives of a set of equations with all of the system variables in consideration. The above equation represents an expansion of x about it's filtered state value approximation $x_{t-1,t-1}$ and x_t is calculated from it assuming no dynamic noise between both the time intervals $t-1$ and t. Similarly, the measurement equation can be written using Taylor series expansion as follows,

$$y_t \approx \tilde{y}_t + \frac{\partial g}{\partial x}(x_t - \tilde{x_t}) + \frac{\partial g}{\partial n}n_t \tag{5}$$

Based on these fundamentals, the steps for EKF have been elaborated in context to the problem statement which also involves a continuous cycle of predict-update.

2.1 Problem statement

The problem represents a sinusoidal motion tracking of an object. The state equations are non-linear in this case mandating the use of the Extended Kalman Filter. If we assume the system states to be as position along the sinusoid x, the rate of position change $\dot{x}$ and the height at a particular position h then the state transition matrix can be written as,

$$f(x_t, a_t) = \begin{bmatrix} x_{t+1} = x_t + \dot{x}_t T \\ \dot{x}_{t+1} = \dot{x}_t + a_t \\ h_{t+1} = \sin \frac{x_t}{10} \end{bmatrix} \tag{6}$$

dynamic noise a_t represents uncertainty in the propagation of the sinusoid over time.

For observations, a sensor detecting the current height of the sinusoid is considered d_t:

$$Y_t = \begin{bmatrix} d_t \end{bmatrix} \tag{7}$$

For this, The observation equations are:

$$g(x_t, n_t) = \begin{bmatrix} d_t = h_t + n_t \end{bmatrix} \tag{8}$$

where n_t is a random sample drawn from $N(0, \sigma_n^2)$ representing measurement noise. To use EKF, we calculate the four Jacobians that come out to be as follows. The derivative of the state transition equations with respect to the state variables is:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial (x, \dot{x}, h)} = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & 0 \\ \frac{1}{10} \cos \frac{x}{10} & 0 & 0 \end{bmatrix} \tag{9}$$

The derivative of the state transition equations with respect to the dynamic noises is:

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial (0, a_t, 0)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{10}$$

The derivative of the observation equations with respect to the state variables is:

$$\frac{\partial g}{\partial x} = \frac{\partial g}{\partial (x, \dot{x}, h)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{11}$$

The derivative of the observation equations with respect to the measurement noises is:

$$\frac{\partial g}{\partial n} = \frac{\partial g}{\partial (n_t)} = \begin{bmatrix} 1 \end{bmatrix} \tag{12}$$

The next part is computing of the covariance matrices. The covariance of the dynamic noises is:

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma_a^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{13}$$

The covariance of the measurement noises is

$$R = \left[\sigma_n^2\right] \tag{14}$$

The system covariance matrix S is a 3x3 matrix whose diagonal elements represent variances of the states as per their distribution and the other elements are covariances which would assume a comparatively lower value.

We then run the loop with steps identical to the Kalman Filter algorithm,

The first step is the prediction of the states for the current timestamp using previous stage's filtered values.

$$X_{3\times1} = f_{3\times1} \tag{15}$$

The second step is the computation of the predicted state covariance matrix

$$S_{3\times3} = \left(\frac{\partial f}{\partial x}\right)_{3\times3} S_{3\times3} \left(\frac{\partial f}{\partial x}\right)^T_{3\times3} + \left(\frac{\partial f}{\partial a}\right)_{3\times3} Q_{3\times3} \left(\frac{\partial f}{\partial a}\right)^T_{3\times3} \tag{16}$$

Then comes the step of calculating the Kalman Gain

$$K_{3\times1} = S_{3\times3} \left(\frac{\partial g}{\partial x}\right)^T_{3\times1} \left[\left(\frac{\partial g}{\partial x}\right)_{1\times3} S_{3\times3} \left(\frac{\partial g}{\partial x}\right)^T_{3\times1} + \left(\frac{\partial g}{\partial n}\right)_{1\times1} R_{1\times1} \left(\frac{\partial g}{\partial n}\right)^T_{1\times1}\right]^{-1} \tag{17}$$

The filtered state and state covariance values are then computed using both the prediction from state transition and measurement data.

$$X_{3\times1} = X_{3\times1} + K_{3\times1}(Y_{1\times1} - g_{1\times1}) \tag{18}$$

$$S_{3\times3} = \left[I_{3\times3} - K_{3\times1} \left(\frac{\partial g}{\partial x}\right)_{1\times3}\right] S_{t,t-1_{3\times3}} \tag{19}$$

3 Observations

3.1 1D constant velocity tracking

The variation of the noises in the system and that of the sensors produce a difference in the way the filter computes the updated values and hence this is something that needs to be determined as per the subjective scenario.

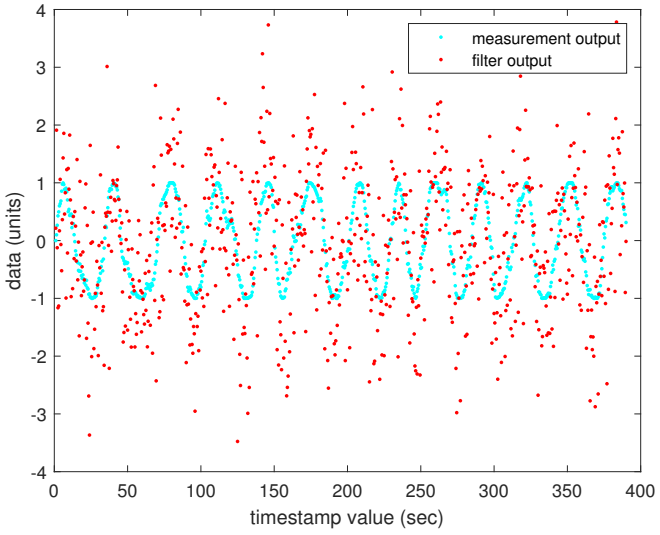

Figure 1: Filter output versus measurement data

For the above figure, the measurement noise $N(0, \sigma_n^2)$ is kept at 0.001 and the dynamic noise$N(0, \sigma_a^2)$ is kept at 0.03.
It can be seen that the updated state values from the filter are dispersed similar to the sensor readings but are quite spread out from each other indicating a weaker trend than the measurement data. This means that the filter takes a weighted consideration of both the measurement and the prediction.

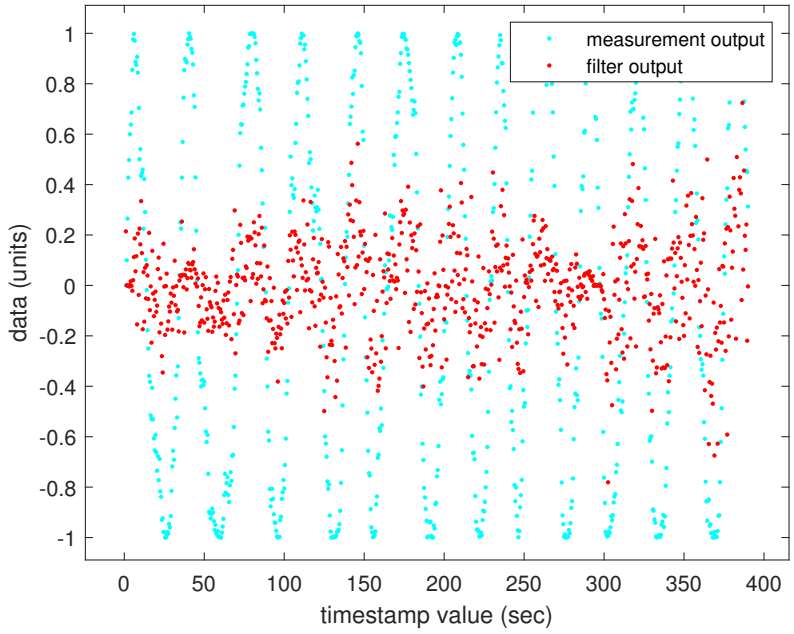

Figure 2: Filter output versus measurement data with more measurement noise

For the above figure, the measurement noise $N(0, \sigma_n^2)$ is increased to 10 and the dynamic noise $N(0, \sigma_a^2)$ is kept at 0.01.
It can be seen that the updated state values from the filter are condensed in a region indicating a clear trend while the measurement data is dispersed sparsely in comparison. This means that the filter outputted values that subsided with the prediction value to a larger extent than the measurement data.

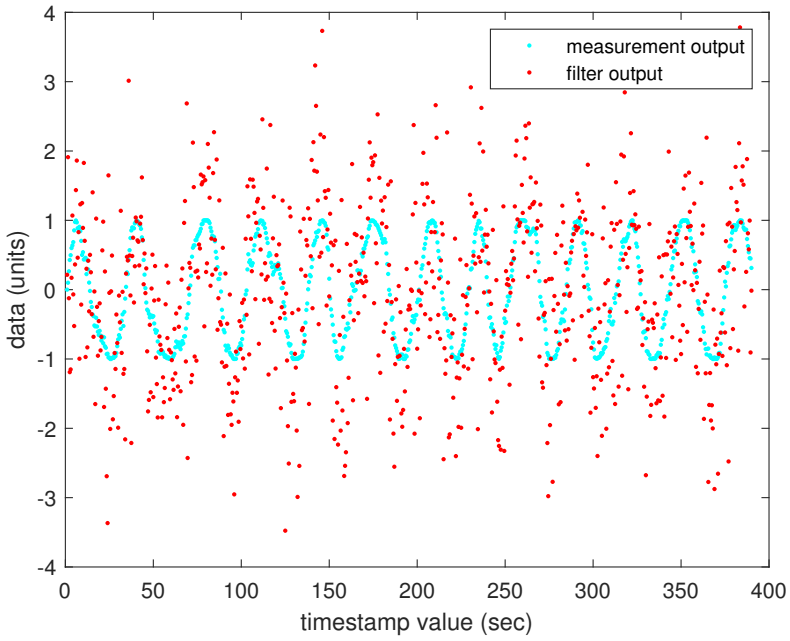

Figure 3: Filter output versus measurement data with more dynamic noise

For the above figure, the measurement noise $N(0, \sigma_n^2)$ is decreased back to 0.0001 and the dynamic noise is increased to $N(0, \sigma_a^2) = 500$.
It can be seen that the updated state values from the filter are condensed in a region indicating a clear trend while the measurement data is dispersed sparsely in comparison. This means that the filter outputted values that subsided with the prediction value to a larger extent than the measurement data.

4 Results and conclusion

The conclusion that can be drawn from the above plots is that there is no true value of any quantity that can be measured. Any estimate is always a weighted component of the sensor reading and the prediction made by the filter. The updated output tends to follow the trends of estimate values for lower ratios of $N(0, \sigma_a^2)/N(0, \sigma_n^2)$ and the trends of the measurement are followed more for the higher values of $N(0, \sigma_a^2)/N(0, \sigma_n^2)$.

5 Appendix

5.1 Code: Sinusoidal tracking EKF

```
clc;
clear;
urlwrite("https://cecas.clemson.edu/~ahoover/ece854/labs/sin-data.txt","measurement_Q1.txt")
%%
inp_filename = "measurement_Q1.txt"
fid = fopen(inp_filename); % open the file
iCtr = 0;
y_q1=[];
true_x_q1=[];
while ~feof(fid) % loop over the following until the end of the file is reached.
  line = fgets(fid); % read in one line
  data1 = strsplit(line);
  true_x_q1(end+1) = str2double(data1(1));
  y_q1(end+1) = str2double(data1(2));
end
  true_x_q1=(true_x_q1)';
  y_q1=(y_q1)';
%%

%assuming a time array corresponding with measurement data with a frequency of 1Hz
time_q1=(1:0.5:length(y_q1)/2);

%assuming a position array corresponding with a predicted state in column 1
%and an updated state in column 2
x_q1=zeros(length(y_q1),2);
x_dot_q1=zeros(length(y_q1),2);
h_q1=zeros(length(y_q1),2);

%assuming initial x and x_dot to be zero and height equal to measurement
x_q1(1,1:end) = 0;
x_dot_q1(1,1:end) = 0;
h_q1(1,1:end) = y_q1(1)*0.9;

%state vector containing both predicted and updated values
X_q1 = [x_q1(1,1:end); x_dot_q1(1,1:end); h_q1(1,1:end)];
X_arr = [];

X_arr{end+1}=X_q1;


noise_x = 0.03;
noise_y = 0.001;
```

```
%for first calculations of f and g, i.e. using the 1 timestamp values to calculate predicted

%    x_q1(t+1)          x_dot(t+1)           h(t+1)
f=zeros(length(y_q1),3);
g = zeros(length(y_q1),1);
S_prev_q1 = [1 0 0; 0 1 0; 0 0 1];
S_next_q1 = 0;
df_da = [0 0 0; 0 1 0; 0 0 0];
dg_dx = [0 0 1];
dg_dn = [1];
Q = [0 0 0; 0 noise_x^2 0; 0 0 0];
R = [noise_y^2];

Kalman_filter(X_q1, X_arr, y_q1, true_x_q1, time_q1, S_prev_q1, S_next_q1, R, Q, df_da, dg_d

%%
%
% Increasing the measurement noise so that the filter relies more on the prediction


%state vector containing both predicted and updated values
X_q1_2 = [x_q1(1,1:end); x_dot_q1(1,1:end); h_q1(1,1:end)];
X_arr_2 = [];

X_arr_2{end+1}=X_q1_2;


noise_x_2 = 0.1e-1;
noise_y_2 = 10;
S_prev_q1 = [1 0 0; 0 1 0; 0 0 1];
S_next_q1 = 0;
df_da = [0 0 0; 0 1 0; 0 0 0];
dg_dx = [0 0 1];
dg_dn = [1];
Q_2 = [0 0 0; 0 noise_x_2^2 0; 0 0 0];
R_2 = [noise_y_2^2];

Kalman_filter(X_q1_2, X_arr_2, y_q1, true_x_q1, time_q1, S_prev_q1, S_next_q1, R_2, Q_2, df_
% Increasing the dynamic noise so that the filter relies more on the measurement


%state vector containing both predicted and updated values
X_q1_3 = [x_q1(1,1:end); x_dot_q1(1,1:end); h_q1(1,1:end)];
X_arr_3 = [];

X_arr_3{end+1}=X_q1_3;
```

```
noise_x_3 = 500;
noise_y_3 = 0.0001;
S_prev_q1 = [1 0 0; 0 1 0; 0 0 1];
S_next_q1 = 0;
df_da = [0 0 0; 0 1 0; 0 0 0];
dg_dx = [0 0 1];
dg_dn = [1];
Q_3 = [0 0 0; 0 noise_x_3^2 0; 0 0 0];
R_3 = [noise_y_3^2];

Kalman_filter(X_q1_3, X_arr_3, y_q1, true_x_q1, time_q1, S_prev_q1, S_next_q1, R_3, Q_3, df_

%%
function Kalman_filter(X_q1, X_arr, y_q1, true_x_q1, time_q1, S_prev_q1, S_next_q1, R, Q, df

    for i=2:1:length(y_q1)-1

        %step 1: Predict next state
        f = [X_q1(1,2) + X_q1(2,2)*(time_q1(i)-time_q1(i-1)); X_q1(2,2) + noise_x; sind(X_q1
        X_q1(1,1)=f(1);
        X_q1(2,1)=f(2);
        X_q1(3,1)=f(3);

        %step 2: predict next state covariance
        df_dx = [1 time_q1(i) 0; 0 1 0; ((1/10)*cosd(f(1,1)/10)) 0 0];
        S_next_q1 = df_dx*S_prev_q1*df_dx' + df_da*Q*df_da';

        %step 3: obtain measurements (there as input)
        g = f(3);

        %step 4: calculate Kalman Gain
        K_q1 = S_next_q1*(dg_dx)'*((dg_dx*S_next_q1*(dg_dx)'+ dg_dn*R*dg_dn')^-1);

        %step 5: update state
        X_upd_q1 = f + K_q1*(y_q1(i)-(g));

        %placing predictions in the state variable array
        X_q1(1,2)=X_upd_q1(1);
        X_q1(2,2)=X_upd_q1(2);
        X_q1(3,2)=X_upd_q1(3);
        X_arr{end+1}=X_q1;

        %step 6: update state covariance
        S_prev_q1 = [eye(3,3) - K_q1*dg_dx]*S_next_q1;
```

```
    end
    display("done");

    X_arr_mat = cell2mat(X_arr);

    figure(fignum)
    plot(time_q1',true_x_q1(1:end-1),'.c')
    ylabel('data (units)');
    xlabel('timestamp value (sec)');
    hold on;
    plot(time_q1',X_arr_mat(3,2:2:end),'.r');
    hold off;
    legend('measurement output','filter output','location','northeast');
end
```